A Storage-Efficient and Robust Private Information Retrieval Scheme Allowing Few Servers

Daniel Augot, Françoise Levy-Dit-Vehel, Abdullatif Shikfa

A Storage-efficient and Robust Private Information Retrieval Scheme allowing few servers

Daniel Augot[1,2], Françoise Levy-dit-Vehel[1,2,3], Abdullatif Shikfa[4]

[1] INRIA
[2] Laboratoire d'informatique de l'École polytechnique
[3] ENSTA ParisTech/U2IS
[4] Alcatel-Lucent

Abstract. Since the concept of locally decodable codes was introduced by Katz and Trevisan in 2000 [11], it is well-known that information theoretically secure private information retrieval schemes can be built using locally decodable codes [15]. In this paper, we construct a Byzantine robust PIR scheme using the multiplicity codes introduced by Kopparty *et al.* [12]. Our main contributions are on the one hand to avoid full replication of the database on each server; this significantly reduces the global redundancy. On the other hand, to have a much lower locality in the PIR context than in the LDC context. This shows that there exists two different notions: LDC-locality and PIR-locality. This is made possible by exploiting geometric properties of multiplicity codes.

1 Introduction

Private information retrieval allows a user to privately retrieve a record of a database, in the sense that the database server does not know which record the user is asking for. The applications of this functionality are numerous. Imagine for instance doctors having to query a company-wide database storing medical for patients, or a police officer wanting to request financial data from the fiscal administration. In both cases, to respect privacy of the patient, or secrecy of the inquiry, it is desirable that the central administration does not know about the queries sent by these users (the doctor or the police officer). A private information retrieval protocol will allow these users to send their queries to the databases, without revealing what they are asking for (either the name of patient, or the name of the suspect under inquiry). Another example is an Internet user who wants to use cloud-based remote storage services, like DropBox, GoogleDrive, CloudMe, hubiC, etc, to store data, and retrieve portion of its data without revealing to these remote services anything about what he is after.

Related work. The problem of Private Information retrieval (PIR) was introduced in 1995 by Chor, Goldreich, Kushilevitz and Sudan [4]. A PIR protocol is a cryptographic protocol the purpose of which is to protect the privacy of

a user accessing a public database via a server, in the sense that it makes it possible for a user to query a particular record of the database without revealing to the server which record he wants to retrieve. We here deal with *information theoretic* PIR, as opposed to *computationally secure* PIR [13]. In an *information theoretic* PIR setting, a server gets no information about the identity of the record of user interest even if it has unlimited computing power: the queries sent to the server must not be correlated to the actual record the user is looking for. In [4] it is shown that when accessing a database located on a single server, to completely guarantee the privacy of the user in an information theoretic sense, one needs to download the entire database, which results in a communication complexity of $O(N)$, N being the bit-size of the database. Thus scenarios have been introduced where the database is replicated across several, say ℓ, servers, and the proposed schemes have communication complexity $O(N^{1/\ell})$, for $\ell \geq 3$. Such multiple-server settings have been investigated since then, and the best communication complexity to date is $N^{O(1/(\log_2 \log_2 N))}$ for 3-server PIR protocols (from matching vector codes construction [14,6]) and $N^{O((\log_2 \log_2 \ell)/\ell \log_2 \ell)}$ for $\ell \geq 3$ [1] .

Beimel and Stahl [2,3] have proposed several robust information theoretic PIR protocols, based on polynomial interpolation, as well as on Shamir's secret sharing scheme. They have built a generic transformation from regular to robust PIR protocols that relies on perfect hash families. They also addressed the Byzantine setting. Recently, Devet, Goldberg and Heninger [5] proposed an Information-Theoretic PIR tolerating the maximum possible number of Byzantine servers. In all these previous proposals, the (encoded or not) database is fully replicated among the servers.

Our contribution. Our main concern is to reduce the global storage overhead. We achieve this by avoiding full replication of the database among the servers. We use multiplicity codes and exploit the geometry of $\mathbb{F}_q^m$ to partition the encoded database (codeword) of bit-size N into q shares of equal size, and distribute them among the servers (one share for one server). This way, we reduce the storage on each server from N bits down to N/q bits, q being the number of servers, while totally preserving the information theoretic security of the PIR protocol. Here $N = \log_2(q^{\sigma q^m}) = \sigma q^m \log_2 q$, with $\sigma = \binom{m+s-1}{m}$, and s is an auxiliary small integer (say $s \leq 6$) used in the construction of multiplicity codes. Given that the code has rate R, the storage overhead of our scheme is thus $\frac{1}{R}$ instead of $\frac{1}{R}\ell$ for schemes with full replication of the encoded database (as in the standard LDC to PIR reduction), ℓ being the number of servers ($\ell = q$ in our scheme). The number of servers is also drastically reduced, from $\sigma(q-1)$ to q, see Fig 3.

The communication complexity in bits (total number of bits sent by the user to all the servers as queries of our protocol) is $(m-1)q\sigma \log_2 q$, and the total number of bits answered by the servers is $q\sigma^2 \log_2 q$. Thus the communication complexity is $(m-1+\sigma)q\sigma \log_2 q$ bits. Putting $\ell = q$ the number of servers, and in contexts where s is small, say $s \leq 6$, this gives a communication complexity of $O(\ell(\log_2 N)^s)$.

Our protocol tolerates $\nu = \lfloor t \rfloor$ byzantine servers, $t = 1/2(q-1-d/s)$, d being the degree of the multiplicity code, in the sense that even if ν out of q servers always answer wrongly, then the database item can still be correctly recovered by the user. Thus our protocol is a ν-Byzantine robust PIR protocol. The property of being robust is a built-in feature of the decoding algorithms that are involved in the process of retrieving the database item.

Organization of the paper. In section 2 we recall the basics of locally decodable and self-correctable codes, private information retrieval schemes, and the link between the two notions; we also set the necessary material and notation to define multiplicity codes, namely Hasse derivatives. Section 3 describes the multiplicity codes [12] as a generalization of Reed Muller codes, and explains their local decoding. Section 4 contains our main ideas: we explain how we use multiplicity codes in a PIR scenario in such a way as to avoid full replication of the encoded database. We also explain how we achieve the Byzantine robustness property of our protocol. We end the paper by numerical tables showing the main features of the codes (rate, locality) for various parameter sizes.

2 Preliminaries

2.1 Locally decodable and locally self-correctable codes

A code in the ambient space is seen as an encoding map, which encodes a message of k symbols on an alphabet Δ into code-vectors, or *codewords* of n symbols on some alphabet Σ (possibly different from Δ). That is, it is a one-to-one map $C : \Delta^k \to \Sigma^n$. The decoding problem is to find codewords close enough to any element y in the ambient space (the "received word" in coding theory language). Formally, given a distance $d()$, code $C \subset \Sigma^n$, for a given $y = (y_1, \ldots, y_n) \in \Sigma^n$, one has to find one, some, or all codewords $c \in C$ such that $d(c, y)$ is small. In our setting, the distance $d(x, y)$ is the Hamming distance which is the number of indices i where $x_i \neq y_i$. A major concern is to build codes with small redundancy, or equivalently, large *rate*, where the rate is $(k \log |\Delta|)/(n \log |\Sigma|)$. In classical settings, $\Delta = \Sigma$, and the rate is simply k/n.

Locally decodable codes, in short LDCs, allow efficient sublinear time decoding. More precisely, an ℓ-query LDC allows to probabilistically recover any symbol of a message by looking at only $\ell \leq k$ randomly chosen coordinates of its - possibly corrupted - encoding. The major objective is to have $\ell \ll k$. Although LDCs appeared in the PCP literature in early 90's [15], their first formal definition is due to Katz and Trevisan in 2000 [11]. The number ℓ of queried symbols is the *query complexity*, that we also call here *locality*. Formally:

Definition 1. *A code $C : \Delta^k \to \Sigma^n$ is* (ℓ, δ)-locally decodable *if there exists a randomized decoding algorithm $\mathcal{A}$ such that*

1. *for any* message $x \in \Delta^k$ *and any $y \in \Sigma^n$ with $d(C(x), y) < \delta n$, we have, for all $i \in [k]$,* $\Pr[\mathcal{A}^y(i) = x_i] \geq \frac{2}{3}$,

2. $\mathcal{A}$ *makes at most* ℓ *queries to* y.

Here, and in the following, $\mathcal{A}^y$ means that $\mathcal{A}$ is given query access to y, and the probability is taken over all internal random coin tosses of $\mathcal{A}$. In the case when one wants to probabilistically recover *any* codeword symbol and not only information symbols, one has the following definition.

Definition 2. *A code* $C : \Delta^k \to \Sigma^n$ *is* (ℓ, δ)-locally self-correctable (LCC) *if there exists a randomized decoding algorithm* $\mathcal{A}$ *such that*

1. *for any* codeword $c \in \Sigma^n$ *and* $y \in \Sigma^n$ *with* $d(c, y) < \delta n$*, we have, for all* $i \in [k]$, $\Pr[\mathcal{A}^y(i) = c_i] \geq \frac{2}{3}$,
2. $\mathcal{A}$ *makes at most* ℓ *queries to* y.

When $\Delta = \Sigma = \mathbb{F}_q$, the finite field with q elements, and when the code is $\mathbb{F}_q$-linear, one can easily construct an LDC from a LCC [16]. No known constructions of LDCs or LCCs minimize both ℓ and the length n simultaneously. The oldest class of LDCs are the Reed-Muller codes over $\mathbb{F}_q$, whose codewords are the evaluations of m-variate polynomials of total degree at most d over $\mathbb{F}_q$ on all the points of $\mathbb{F}_q^m$. The main issues are thus to minimize one parameter given that the other one is fixed. With this respect, constructions of subexponential length codes with constant query complexity $\ell \geq 3$ exist [15]. On the other side, constant rate LDCs feature an ℓ which is known to lie between $\Omega(\log_2 k)$ and $\Theta(k^\epsilon)$, with explicit constructions for the latter bound. A major result is the construction of high-rate (i.e. $> 1/2$) locally self-correctable codes with sublinear query complexity, in the presence of a constant (as a function of the distance of the code) fraction of errors. Those codes are known as *Multiplicity Codes* and were introduced by Kopparty, Saraf and Yekhanin in 2011 [12]. They generalize the Reed-Muller codes by evaluating high degree multivariate polynomials as well as their partial derivatives up to some order s. Using high-degree polynomials improves on the rate, while evaluating their partial derivatives compensates for the loss in distance. Other LDC constructions achieving rate $> 1/2$ and query complexity n^ϵ are the one of Guo *et al.* [8] based on lifting affine-invariant codes (namely, Reed-Solomon codes), and the Expander codes of Hemenway *et al.* [10].

In this work, we use Multiplicity codes, but recall Reed-Muller codes and their local decoding for the sake of comprehension. These codes provide the simplest geometric setting for partitioning a codeword and laying it out on servers. We think such a partition can be done for other families of LDC codes, e.g. matching-vector codes, affine invariant codes and possibly Expander codes.

2.2 Private information retrieval schemes

We model the database as a string x of length k over Δ. An ℓ-server PIR scheme involves ℓ servers $S_1, \ldots, S_\ell$, each holding the same database x, and a user who knows k and wants to retrieve some value x_i, $i \in [k]$, without revealing any information about i to the servers.

Definition 3 (Private Information Retrieval (PIR)). *An ℓ-server p-PIR protocol is a triple $(\mathcal{Q}, \mathcal{A}, \mathcal{R})$ of algorithms running as follows:*

1. *User obtains a random string s; then he invokes $\mathcal{Q}$ to generate an ℓ-tuple of queries $(q_1, \ldots, q_\ell) = \mathcal{Q}(i, s)$.*
2. *For $1 \leq j \leq \ell$, User sends q_j to server S_j;*
3. *Each S_j answers $a_j = \mathcal{A}(j, x, q_j)$ to User;*
4. *User recovers x_i by applying the reconstruction algorithm $\mathcal{R}(a_1, \ldots, a_\ell, i, s)$.*

Furthermore the protocol has the Correctness property*: for any $x \in \Delta^k$, $i \in [k]$, User recovers x_i with probability at least p; and the* Privacy property*: each server individually can obtain no information about i.*

The Privacy property can be obtained by requiring that for all $j \in [\ell]$, the distribution of the random variables $\mathcal{Q}(i, \cdot)_j$ are identical for all $i \in [k]$. Katz and Trevisan [11], introduced a notion very relevant in the context of locally decodable codes: that of *smooth codes*. The notion of smooth codes captures the idea that a decoder cannot read the same index too often, and implies that the distributions $\mathcal{Q}(i, \cdot)_j$ are close to uniform. All known examples are such that the distribution $\mathcal{Q}(i, \cdot)_j$ are actually uniform. Uniform distribution of the queries among codeword (or received word) coordinates is what is needed in the PIR setting in order to achieve information theoretic privacy of the queries. The locality as a core feature of LDCs, together with the fact that in all known constructions of LDCs the queries made by the local decoding algorithm $\mathcal{A}$ are uniformly distributed, make the application of LDCs to PIR schemes quite natural. Note also that conversely PIR schemes can be used to build LDCs with best asymptotic code-lengths [1,14,6]. The lemma below describes how it formally works.

Lemma 1 (Application of LDCs to PIR schemes). *Suppose there exists an ℓ-query locally decodable code $C : \Delta^k \to \Sigma^n$, in which each decoder's query is uniformly distributed over the set of codeword coordinates. Then there exists an ℓ-server 1-PIR protocol with $O(\ell(\log_2 n + \log_2 |\Sigma|))$ communication to access a database $x \in \Delta^k$.*

Proof. Given an LDC $C : \Delta^k \to \Sigma^n$ as in the lemma, one constructs the following PIR protocol. First, in a preprocessing step, for $1 \leq j \leq \ell$, server S_j encodes x with C. Then, to actually run the protocol, User tosses random coins and invokes the local decoding algorithm to determine the queries $(q_1, \ldots, q_\ell) \in [n]^\ell$ such that x_i can be computed from $\{C(x)_{q_j}\}_{1 \leq j \leq \ell}$. For $1 \leq j \leq \ell$, User sends $q_j \in [n]$ to server S_j, and each server S_j answers $C(x)_{q_j} \in \Sigma$. Finally, User applies the local decoding algorithm of C to recover x_i.

This protocol has the communication complexity claimed in the lemma. Furthermore, as the user applies the local decoding algorithm with non corrupted inputs $\{C(x)_{q_j}\}_{1 \leq j \leq \ell}$, he retrieves x_i with probability 1. Uniformity of the distribution of the decoder's queries over $[n]$ ensures the information-theoretic privacy of the protocol.

2.3 Hasse derivative for multivariate polynomials

Notation Considering m indeterminates $X_1, \ldots, X_m$, and m positive integers $i_1, \ldots, i_m$, we use the short-hand notation

$$\begin{array}{lll} \boldsymbol{X} = (X_1, \ldots, X_m) & \boldsymbol{X}^{\boldsymbol{i}} = X_1^{i_1} \cdots X_m^{i_m}, & \mathbb{F}_q[\boldsymbol{X}] = \mathbb{F}_q[X_1, \ldots, X_m] \\ \boldsymbol{i} = (i_1, \ldots, i_m) \in \mathbb{N}^m & |\boldsymbol{i}| = i_1 + \cdots + i_m & \boldsymbol{P} = (p_1, \ldots, p_m) \in \mathbb{F}_q^m \end{array}$$

i.e. we use bold symbols for vectors, points, etc, and standard symbols for unidimensional scalars, variables, etc. In general, we write polynomials $Q \in \mathbb{F}_q[\boldsymbol{X}] = \mathbb{F}_q[X_1, \ldots, X_m]$ without parenthesis and without variables, and $Q(\boldsymbol{X})$ (resp. $Q(\boldsymbol{P})$) when the evaluation on indeterminates (resp. points) has to be specified. For $\boldsymbol{i}, \boldsymbol{j} \in \mathbb{N}^m$, $\boldsymbol{i} \gg \boldsymbol{j}$ means $i_t \geq j_t \, \forall 1 \leq t \leq m$.

Hasse derivative Given a multi-index $\boldsymbol{i}$, and $F \in \mathbb{F}_q[\boldsymbol{X}]$, the $\boldsymbol{i}$-th Hasse derivative of F, denoted by $H(F, \boldsymbol{i})$, is the coefficient of $\boldsymbol{Z}^{\boldsymbol{i}}$ in the polynomial $F(\boldsymbol{X} + \boldsymbol{Z}) \in \mathbb{F}_q[\boldsymbol{X}, \boldsymbol{Z}]$, where $\boldsymbol{Z} = (Z_1, \ldots, Z_m)$. More specifically, let $F(\boldsymbol{X}) = \sum_{\boldsymbol{j} \gg 0} f_{\boldsymbol{j}} \boldsymbol{X}^{\boldsymbol{j}}$, then

$$F(\boldsymbol{X} + \boldsymbol{Z}) = \sum_{\boldsymbol{j}} f_{\boldsymbol{j}} (\boldsymbol{X} + \boldsymbol{Z})^{\boldsymbol{j}} = \sum_{\boldsymbol{i}} H(F, \boldsymbol{i})(\boldsymbol{X}) \boldsymbol{Z}^{\boldsymbol{i}},$$

where $\boldsymbol{Z}^{\boldsymbol{i}}$ stands for $Z_1^{i_1} \cdots Z_m^{i_m}$, and

$$H(F, \boldsymbol{i})(\boldsymbol{X}) = \sum_{\boldsymbol{j} \gg \boldsymbol{i}} f_{\boldsymbol{j}} \binom{\boldsymbol{j}}{\boldsymbol{i}} \boldsymbol{X}^{\boldsymbol{j} - \boldsymbol{i}} \quad \text{with } \binom{\boldsymbol{j}}{\boldsymbol{i}} = \binom{j_1}{i_1} \cdots \binom{j_m}{i_m}.$$

Considering a vector $\boldsymbol{V} \in \mathbb{F}_q^m \setminus \{0\}$, and a base point $\boldsymbol{P}$, we consider the restriction of F to the line $D = \{\boldsymbol{P} + t\boldsymbol{V} : \, t \in \mathbb{F}_q\}$, which is a univariate polynomial that we denote by $F_{\boldsymbol{P}, \boldsymbol{V}}(T) = F(\boldsymbol{P} + T\boldsymbol{V}) \in \mathbb{F}_q[T]$. We have the following relations:

$$F_{\boldsymbol{P}, \boldsymbol{V}}(T) = \sum_{\boldsymbol{j}} H(F, \boldsymbol{j})(\boldsymbol{P}) \boldsymbol{V}^{\boldsymbol{j}} T^{|\boldsymbol{j}|}, \tag{1}$$

$$\text{coeff}(F_{\boldsymbol{P}, \boldsymbol{V}}, i) = \sum_{|\boldsymbol{j}| = i} H(F, \boldsymbol{j})(\boldsymbol{P}) \boldsymbol{V}^{\boldsymbol{j}}, \tag{2}$$

$$H(F_{\boldsymbol{P}, \boldsymbol{V}}, i)(\alpha) = \sum_{|\boldsymbol{j}| = i} H(F, \boldsymbol{j})(\boldsymbol{P} + \alpha \boldsymbol{V}) \boldsymbol{V}^{\boldsymbol{j}}, \quad \text{for all } \alpha \in \mathbb{F}_q \tag{3}$$

3 Multiplicity codes

3.1 Local decoding of Reed-Muller codes

We enumarte the finite field $\mathbb{F}_q$ with q elements as $\mathbb{F}_q = \{\alpha_0 = 0, \alpha_1, \ldots, \alpha_{q-1}\}$. We denote by $\mathbb{F}_q[\boldsymbol{X}]_d$ the set of polynomials of degree less than or equal to d, which has dimension $k = \binom{m+d}{d}$. We enumerate all the points in $\mathbb{F}_q^m$:

$$\mathbb{F}_q^m = \{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\} \tag{4}$$

where $\boldsymbol{P}_i = (P_{i,1}, \ldots, P_{i,m}) \in \mathbb{F}_q^m$, is an m-tuple of $\mathbb{F}_q$-symbols, and $n = q^m$. We encode a polynomial F of degree $\leq d$ into a codeword c of length n using the evaluation map

$$\begin{aligned} \mathrm{ev} : \mathbb{F}_q[\boldsymbol{X}]_d &\to \mathbb{F}_q^n \\ F &\mapsto (F(\boldsymbol{P}_1), \ldots, F(\boldsymbol{P}_n)) \end{aligned}$$

and the d-th order Reed-Muller code is $\mathrm{RM}_d = \{\mathrm{ev}(F) \mid F \in \mathbb{F}_q[\boldsymbol{X}]_d\}$. The evaluation map ev encodes k symbols into n symbols, and the rate is $R = k/n \in [0, 1]$. A codeword $c \in RM_d$ can be indexed by integers as $c = (c_1, \ldots, c_n)$ or by points as $c = (c_{\boldsymbol{P}_1}, \ldots, c_{\boldsymbol{P}_n})$, where $c_i = c_{\boldsymbol{P}_i}$.

Assuming $d < q$, we now recall how RM_d achieves a locality of $\ell = q - 1$ as follows. Suppose that $c = \mathrm{ev}(F) \in \mathrm{RM}_d$ is a codeword, and that $c_j = c_{\boldsymbol{P}_j}$ is looked for. Then, the local decoding algorithm randomly picks a non-zero vector $\boldsymbol{V} \subset \mathbb{F}_q^m \setminus \{0\}$ and considers the line D of direction $\boldsymbol{V}$ passing through $\boldsymbol{P}_j$:

$$\begin{aligned} D &= \{\boldsymbol{P}_j + t \cdot \boldsymbol{V} \mid t \in \mathbb{F}_q\} = \{\boldsymbol{P}_j + 0 \cdot \boldsymbol{V}, \boldsymbol{P}_j + \alpha_1 \cdot \boldsymbol{V}, \ldots, \boldsymbol{P}_j + \alpha_{q-1} \cdot \boldsymbol{V}\} \\ &= \{\boldsymbol{R}_0 = \boldsymbol{P}_j, \ldots, \boldsymbol{R}_{q-1}\} \subset \mathbb{F}_q^m. \end{aligned}$$

Then, the points $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_{q-1}$ are sent as queries, and the decoding algorithm receives the answer:

$$\left(y_{\boldsymbol{R}_1}, \ldots, y_{\boldsymbol{R}_{q-1}}\right) \in \mathbb{F}_q^{q-1}.$$

In case of no errors, $\left(y_{\boldsymbol{R}_1}, \ldots, y_{\boldsymbol{R}_{q-1}}\right) = \left(c_{\boldsymbol{R}_1}, \ldots, c_{\boldsymbol{R}_{q-1}}\right)$. Now

$$c_{\boldsymbol{R}_u} = F(\boldsymbol{P}_j + \alpha_u \cdot \boldsymbol{V}) = F_{\boldsymbol{P},\boldsymbol{V}}(\alpha_u),\ u = 1, \ldots, q-1,$$

where

$$F_{\boldsymbol{P},\boldsymbol{V}} = F(\boldsymbol{P} + T \cdot \boldsymbol{V}) \in \mathbb{F}_q[T] \tag{5}$$

is the restriction of F to the line D, which is a univariate polynomial of degree less than or equal to d. That is, $\left(c_{\boldsymbol{R}_1}, \ldots, c_{\boldsymbol{R}_{q-1}}\right)$ belongs to a Reed-Solomon code RS_d of length $q-1$ and dimension $d+1$. In case of errors, $\left(y_{\boldsymbol{R}_1}, \ldots, y_{\boldsymbol{R}_{q-1}}\right)$ is a noisy version of it. Using a decoding algorithm of RS_d, one can recover $F_{\boldsymbol{P},\boldsymbol{V}}$, and then $c_{\boldsymbol{P}_j}$ is found as $c_{\boldsymbol{P}_j} = F_{\boldsymbol{P},\boldsymbol{V}}(0)$.

The main drawback of these codes is the condition $d < q$, which imposes a dimension $k = \binom{d+m}{m} < \binom{q+m}{m} \sim q^m/m!$. For a fixed alphabet $\mathbb{F}_q$, the rate $R = k/q^m < 1/m!$ goes to zero very fast when the codes get longer.

3.2 Multiplicity codes and their local decoding

To obtain codes with higher rates, we need a derivation order $s > 0$ and an extended notion of evaluation. There are $\sigma = \binom{m+s-1}{m}$ Hasse derivatives $H(F, \boldsymbol{i})$ of a polynomial F for multi-indices $\boldsymbol{i}$ such that $|\boldsymbol{i}| < s$. Letting $\Sigma = \mathbb{F}_q^\sigma$, we generalize the evaluation map at a point $\boldsymbol{P}$:

$$\begin{aligned} \mathrm{ev}_{\boldsymbol{P}}^s : \mathbb{F}_q[\boldsymbol{X}] &\to \mathbb{F}_q^\sigma \\ F &\mapsto (H(F, \boldsymbol{v})(\boldsymbol{P}))_{|\boldsymbol{v}|<s} \end{aligned}$$

and, given an enumeration of the points as in Eq. 4, the total evaluation rule is

$$\begin{aligned}\mathrm{ev}^s : \mathbb{F}_q[\boldsymbol{X}] &\to \Sigma^n \\ F &\mapsto \left(\mathrm{ev}^s_{\boldsymbol{P}_1}(F), \ldots, \mathrm{ev}^s_{\boldsymbol{P}_n}(F)\right).\end{aligned}$$

Given $y = \mathrm{ev}^s_{\boldsymbol{P}}(F) \in \Sigma$, we denote by $y_{\boldsymbol{v}}$ the coordinate of y corresponding to the $\boldsymbol{v}$-th derivative of F. As in the case of classical Reed-Muller codes, we denote by $(c_1, \ldots, c_n) = (c_{\boldsymbol{P}_1}, \ldots, c_{\boldsymbol{P}_n}) = \mathrm{ev}^s(F)$, i.e. $c_i = c_{\boldsymbol{P}_i} = \mathrm{ev}^s_{\boldsymbol{P}_i}(F)$. We can consider $\mathbb{F}_q[\boldsymbol{X}]_d$, with $d < s(q-1)$ [12], and the corresponding code is

$$\mathrm{Mult}^s_d = \{\mathrm{ev}^s(F) \mid F \in \mathbb{F}_q[\boldsymbol{X}]_d\}.$$

Using the language of locally decodable codes, we have a code $\mathrm{Mult}^s_d : \Delta^k \to \Sigma^n$, with $\Delta = \mathbb{F}_q$, and $\Sigma = \mathbb{F}_q^\sigma$. The code Mult^s_d, is a $\mathbb{F}_q$-linear space, whose dimension over $\mathbb{F}_q$ is $k = \binom{m+d}{d}$. Its rate is $R = (\log_q |\mathbb{F}_q[X]_d|)/(\log_q |\Sigma^n|) = k/(\sigma n) = \binom{m+d}{m} / \left(\binom{m+s-1}{m} \cdot q^m\right)$. Its minimum distance is (from Generalized Schwartz-Zippel Lemma) $q^m - \frac{d}{s} q^{m-1}$.

This family of codes has a locality of $(q-1)\sigma = (q-1)\binom{m+s-1}{m}$ queries. Here is how the local decoding algorithm works. Let j be the index of the point where we want to local decode, i.e. $c_j = c_{\boldsymbol{P}_j}$ is looked for. The algorithm randomly picks σ vectors $\boldsymbol{U}_i \in \mathbb{F}_q^m \setminus \{0\}$, $i = 1, \ldots, \sigma$. For each $\boldsymbol{U}_i$, $i = 1, \ldots, \sigma$, consider the line of direction $\boldsymbol{U}_i$ passing through $\boldsymbol{P}_j$:

$$\begin{aligned}D_i &= \{\boldsymbol{P}_j + 0 \cdot \boldsymbol{U}_i, \boldsymbol{P}_j + \alpha_1 \cdot \boldsymbol{U}_i, \ldots, \boldsymbol{P}_j + \alpha_{q-1} \cdot \boldsymbol{U}_i\} \\ &= \{\boldsymbol{R}_{i,0} = \boldsymbol{P}_j, \boldsymbol{R}_{i,1}, \ldots, \boldsymbol{R}_{i,q-1}\} \subset \mathbb{F}_q^m\end{aligned}$$

For each i, $1 \le i \le \sigma$, the algorithm queries the received word at points $\boldsymbol{R}_{i,1}, \ldots, \boldsymbol{R}_{i,q-1}$, and gets the answers

$$\left(y_{\boldsymbol{R}_{i,1}}, \ldots, y_{\boldsymbol{R}_{i,q-1}}\right) \in \Sigma^{q-1},$$

thus a total of $(q-1)\sigma$ queries in $\mathbb{F}_q^m$, and $\sigma(q-1)$ answers from Σ. In case of no errors, we have

$$(y_{\boldsymbol{R}_{i,b}})_{\boldsymbol{v}} = H(F, \boldsymbol{v})(\boldsymbol{R}_{i,b}), \quad b = 1, \ldots, q-1,$$

where $(y_{\boldsymbol{R}_{i,b}})_{\boldsymbol{v}}$ is the $\boldsymbol{v}$-th coordinate of $y_{\boldsymbol{R}_{i,b}}$, and, using Eq. 3, we can compute

$$H(F_{\boldsymbol{P}_j, \boldsymbol{U}_i}, e)(\alpha_b) = \sum_{|\boldsymbol{v}|=e} H(F, \boldsymbol{v})(\boldsymbol{R}_{i,b}) \boldsymbol{U}_i^{\boldsymbol{v}} \quad \begin{cases} 1 \le b \le q-1, \\ 0 \le e < s \end{cases} \tag{6}$$

Having the values $H(F_{\boldsymbol{P}_j, \boldsymbol{U}_i}, e)(\alpha_b)$, for $1 \le b \le q-1$ and $|\boldsymbol{v}| < s$, we can then recover $F_{\boldsymbol{P}_j, \boldsymbol{U}_i}$ by Hermite interpolation. Next we solve, for the indeterminates $H(F, \boldsymbol{v})(\boldsymbol{P}_j)$, $|\boldsymbol{v}| < s$, the linear system derived from Eq. 2:

$$\mathrm{coeff}(F_{\boldsymbol{P}_j, \boldsymbol{U}_i}, e) = \sum_{|\boldsymbol{v}|=e} H(F, \boldsymbol{v})(\boldsymbol{P}_j) \boldsymbol{U}_i^{\boldsymbol{v}} \quad \begin{cases} e = 0, \ldots, s-1, \\ i = 1, \ldots \sigma, \end{cases}$$

and we output $\{H(F,\boldsymbol{v})(\boldsymbol{P}_j),\ |\boldsymbol{v}|<s\} = \mathrm{ev}^s_{\boldsymbol{P}_j}(F)$.

In case of errors, for each direction $\boldsymbol{U}_i$, we define a function $h_i : \mathbb{F}_q^* \to \mathbb{F}_q^{\{0,\dots,s-1\}}$, $\alpha_b \mapsto h_i(\alpha_b)$, such that

$$(h_i(\alpha_b))(e) = \sum_{|\boldsymbol{v}|=e} (y_{\boldsymbol{R}_{i,b}})_{\boldsymbol{v}} \boldsymbol{U}_i^{\boldsymbol{v}}, \quad \begin{cases} 1 \le b \le q-1 \\ 0 \le e < s \end{cases} \tag{7}$$

By virtue of Eq. 6, note that $h_i(\alpha_b)(e)$ is the (erroneous) e-th Hasse derivative of $F_{\boldsymbol{P}_j,\boldsymbol{U}_i}$ at α_b.

Having $h_i(\alpha_b)(e)$ for all $e \in \{0,\dots,s-1\}$ and all $b \in \{1,\dots,q-1\}$, $F_{\boldsymbol{P}_j,\boldsymbol{U}_i}$ is recovered using a decoding algorithm of univariate multiplicity codes (see [12]), provided $d(\mathrm{ev}^s(F_{\boldsymbol{P}_j,\boldsymbol{U}_i}), h_i) \le \frac{(q-1)-d/s}{2}$. Once we have recovered $F_{\boldsymbol{P}_j,\boldsymbol{U}_i}$, we solve for the indeterminates $H(F,\boldsymbol{v})(\boldsymbol{P}_j)$, $|\boldsymbol{v}| < s$, the linear system derived from Eq. 2:

$$\mathrm{coeff}(F_{\boldsymbol{P}_j,\boldsymbol{U}_i}, e) = \sum_{|\boldsymbol{v}|=e} H(F,\boldsymbol{v})(\boldsymbol{P}_j)\boldsymbol{U}_i^{\boldsymbol{v}} \quad \begin{cases} t = 0,\dots,s-1, \\ i = 1,\dots\sigma \end{cases} \tag{8}$$

and we output $\{H(F,\boldsymbol{v})(\boldsymbol{P}_j),\ |\boldsymbol{v}|<s\} = \mathrm{ev}^s_{\boldsymbol{P}_j}(F)$. This local decoding algorithm is sketched in Alg 1. In case of more than $\frac{(q-1)-d/s}{2}$ errors in some directions, the linear system 8 may have erroneous equations. In this case, due to lack of space, we refer the reader to [12].

Algorithm 1 Local decoding algorithm for Multiplicity Codes

Require: Oracle Access to $y = (y_1,\dots,y_n)$, a noisy version of $c = \mathrm{ev}^s(F) \in \mathrm{Mult}_d$.
Input: $j \in [n]$, the index of the symbol c_j looked for in c
Output: $c_j = c_{\boldsymbol{P}_j} = \mathrm{ev}^s_{\boldsymbol{P}_j}(F)$

1: Pick distinct σ non zero random vectors $\boldsymbol{U}_1,\dots,\boldsymbol{U}_\sigma$ giving σ different lines
2: **for** i=1 to σ **do**
3: Consider the line

$$D_i = \{\boldsymbol{P}_j + 0\cdot\boldsymbol{U}_i, \boldsymbol{P}_j + \alpha_1\cdot\boldsymbol{U}_i, \dots, \boldsymbol{P}_j + \alpha_{q-1}\cdot\boldsymbol{U}_i\} = \{\boldsymbol{R}_{i,0},\dots,\boldsymbol{R}_{i,q-1}\}$$

4: Send $\boldsymbol{R}_{i,1},\dots,\boldsymbol{R}_{i,q-1}$, as queries,
5: Receive the answers: $y_{\boldsymbol{R}_{i,1}},\dots,y_{\boldsymbol{R}_{i,q-1}}$, $y_{\boldsymbol{R}_{i,b}} \in \mathbb{F}_q^\sigma$.
6: Recover $F_{\boldsymbol{P}_j,\boldsymbol{U}_i}$ from $(y_{\boldsymbol{R}_{i,1}},\dots,y_{\boldsymbol{R}_{i,q-1}})$ using a univariate decoding algorithm on the values $(h_i(\alpha_b))(e)$ defined in Eq. 7.
7: **end for**
8: Solve for the indeterminates $H(F,\boldsymbol{v})(\boldsymbol{P}_j)$, $|\boldsymbol{v}|<s$, the linear system 8.
9: **return** $\{H(F,\boldsymbol{v})(\boldsymbol{P}_j),\ |\boldsymbol{v}|<s\} = \mathrm{ev}^s_{\boldsymbol{P}_j}(F)$.

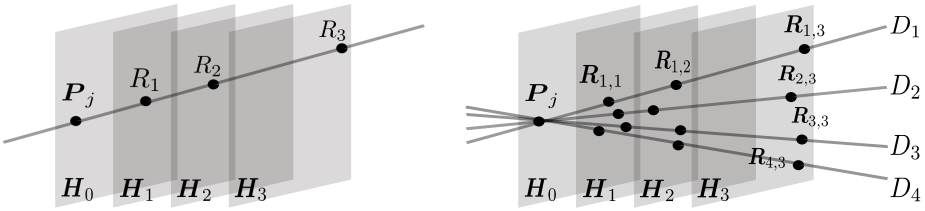

Fig. 1. Transversal lines for simple Reed-Muller codes (a), for Multiplicity Codes (b), assuming that the point $\boldsymbol{P}_j$ corresponding to query j lies on the H_0 hyperplane. Parameters are $q = 4$, $m = 3$, $s = 2$, $\sigma = 4$. Not all point names are displayed for readability.

4 Hyperplane partitions and their use in PIRs

4.1 Affine hyperplanes and servers

Considering Mult_d^s, we show how to equally share a codeword

$$c = \mathrm{ev}^s(f) = \left(\mathrm{ev}^s_{\boldsymbol{P}_1}(f), \ldots, \mathrm{ev}^s_{\boldsymbol{P}_n}(f)\right)$$

on $\ell = q$ servers, using the geometry of $\mathbb{F}_q^m$. This is done as follows: consider H a $\mathbb{F}_q$-linear subspace of $\mathbb{F}_q^m$ of dimension $m-1$. It can be seen as the kernel of a linear map

$$\begin{aligned} f_H : \quad & \mathbb{F}_q^m \to \mathbb{F}_q \\ & (x_1, \ldots, x_m) \mapsto h_1 x_1 + \cdots h_m x_m \end{aligned}$$

for some $(h_1, \ldots, h_m) \in \mathbb{F}_q^m \setminus \{0\}$. Now $\mathbb{F}_q^m$ can be split as the disjoint union of affine hyperplanes $\mathbb{F}_q^m = H_0 \cup H_1 \cup \cdots \cup H_{q-1}$, where

$$H_i = \left\{ \boldsymbol{P} \in \mathbb{F}_q^m \;\mid\; f_H(\boldsymbol{P}) = \alpha_i \right\}, \quad i = 0, \ldots, q-1.$$

As a simple example, consider the $\mathbb{F}_q$-linear hyperplane H of $\mathbb{F}_q^m$:

$$H = \{P = (x_1, \ldots, x_m) \mid x_m = 0\}.$$

Then we have $\mathbb{F}_q^m = H_0 \cup H_1 \cup \ldots H_{q-1}$ where

$$H_i = \left\{ \boldsymbol{P} = (x_1, \ldots, x_m) \in \mathbb{F}_q^m \mid x_m = \alpha_i \right\}, \quad i = 0, \ldots, q-1.$$

Up to a permutation of the indices, we can write any codeword $c = \left(c_{H_0} | \cdots | c_{H_{q-1}}\right)$, where

$$c_{H_i} = \left(\mathrm{ev}^s_{\boldsymbol{P}}(f)\right)_{\boldsymbol{P} \in H_i}, \; i = 0 \ldots, q-1.$$

Now consider an affine line, which is *transversal* to all the hyperplanes. It is a line which can be given by any direction $\boldsymbol{U} \in \mathbb{F}_q^m \setminus \{0\}$ such that $f_H(\boldsymbol{U}) \neq 0$, and which contains a point $\boldsymbol{P}$:

$$D = \{\boldsymbol{P} + t \cdot \boldsymbol{U} \mid t \in \mathbb{F}_q\}.$$

In other words, it is a line not contained in any of the hyperplane $H_0, \ldots, H_{q-1}$. Then,

$$D \cap H_j = \{\boldsymbol{Q}_j\}, \quad j = 0, \ldots, q-1,$$

for some points $\boldsymbol{Q}_0, \ldots, \boldsymbol{Q}_{q-1}$. Now, as long as $\boldsymbol{U}_i$, $i = 1, \ldots, \sigma$, does not belong to H, Algorithm 1 works, using the points $\{\boldsymbol{Q}_{i,j}\}_{0 \le j \le q-1}$, where $D_i \cap H_j = \{\boldsymbol{Q}_{i,j}\}$, D_i being the line with direction $\boldsymbol{U}_i$ passing through $\boldsymbol{P}_j$, one query being a fake one (see section 4.2 below).

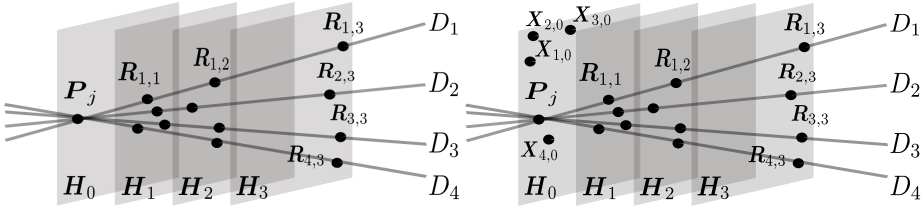

Fig. 2. Parameters are $q = 4$, $m = 3$, $s = 2$, $\sigma = 4$. Queries for a Multiplicity code used as an LDC codes (a), used in PIR scheme (b), assuming that the point $\boldsymbol{P}_j$ corresponding to query j lies on the H_0 hyperplane. In the PIR scheme, random points $\boldsymbol{X}_{1,0}, \ldots, \boldsymbol{X}_{4,0}$ are sent to the server S_0 to hide him the fact that he hosts the index of the request. Not all point names are displayed for readability.

4.2 Use in PIR schemes

Given $\mathbb{F}_q^m = H_0 \cup H_1 \cup \ldots H_{q-1}$, the PIR scheme can be built by requiring that, for $i = 1, \ldots, q$, Server S_i is given c_{H_i} to store. Local decoding must be done using transversal lines. The user will first select σ transversal lines D_i, $i = 1, \ldots, \sigma$, which passes through the point $\boldsymbol{P}_j$ which corresponds to the requested symbol, and query each server S_i at the point $D \cap H_i$. In algorithms 1, 2, the main and only change is to make sure that all lines under consideration are indeed transversal to the chosen hyperplanes. We here explain how this works: the code requires $(q-1)$ queries along each line. In our context, when $\boldsymbol{P}_j$ is requested, all σ lines have to pass through $\boldsymbol{P}_j$. For a direction $\boldsymbol{U}_i$, the queries sent to the servers correspond to $q-1$ points on the line D_i defined by $\boldsymbol{U}_i$, those points being all different from $\boldsymbol{P}_j$. Assume for instance that $\boldsymbol{P}_j = (x_1, \ldots, x_m)$ with $x_m = \alpha_u$, for some u. Query $\boldsymbol{P}_j$ must not be sent to server S_u who stores the c_{H_u} part of the encoded word: S_u would then know that it has the index of the requested coordinate among its possibly queried indices. A solution to this problem is to send σ fake (i.e. random) queries $X_{i,u}$, $i = 1, \ldots, \sigma$, to server S_u, see Fig 2. This is enough to obfuscate server S_u. See Algorithm 2.

Algorithm 2 PIR Protocol from transversal lines on hyperplanes

Preprocessing Phase: The user:

1: chooses q, m, d, s so that the original data x of bit-size k can be encoded using $\mathrm{Mult}_d^s(q)$, i.e. parameters such that $\binom{m+d}{d} \log_2 q \geq k$;
2: encodes the data x into the codeword $c = \mathrm{ev}^s(F)$, where the coefficients of F represent the original data x;
3: sends each server S_ℓ the c_{H_ℓ} part of the codeword.

Online Protocol: To recover $c_j = ev^s_{\boldsymbol{P}_j}(F)$ for an index $j \in [n]$, the user:

4: User selects σ distinct lines $D_i, 1 \leq i \leq \sigma$, transversal to the hyperplanes, and passing through $\boldsymbol{P}_j$;
5: Let ℓ_j be such that $D_i \cap H_{\ell_j} = \boldsymbol{P}_j$, $i = 1, \ldots, \sigma$
6: For $1 \leq \ell \leq q$, $\ell \neq \ell_j$, user sends the queries $\{D_i \cap H_\ell = R_{i,\ell}\}_{1 \leq i \leq \sigma}$ to server ℓ.
7: User sends σ random queries X_{i,ℓ_j}, $i = 1, \ldots, \sigma$ to server S_{ℓ_j};
8: For $1 \leq \ell \leq q$, server sends the answers $\{y_{R_{i,\ell}}\}_{1 \leq i \leq \sigma}$. Answers $\{y_{R_{i,u}}\}_{1 \leq i \leq \sigma}$ are discarded by the user.
9: User then proceeds as in steps 5 to 8 of algorithm 1 to retrieve $\mathrm{ev}^s_{\boldsymbol{P}_j}(F)$.

5 Analysis of the protocol given in algorithm 2

5.1 Overall storage overhead

The natural reduction from locally decodable codes to information theoretically secure private information retrieval schemes leads to two overheads: the first one is $1/R$ where R is the rate of the code used for encoding the data, the second one is ℓ, where ℓ is the number of servers. The total overhead is thus $\ell \cdot 1/R$. Our scheme has an overhead of only $1/R$, which is the natural overhead of the code. With respect to the amount of storage required in each server for encoding k symbols, only k/Rq symbols are required per server. In particular, when $R \geq 1/q$, each server stores less than k symbols, which is the amount of information without redundancy.

5.2 Communication complexity

We count the communication complexity in terms of the number of exchanged bits during the online protocol, discounting the preprocessing phase. The user has to send σ points to each server S_j, $j = 1, \ldots, q$. A point consists in m coordinates in $\mathbb{F}_q$, but since it belongs to an hyperplane, it can be specified with $(m-1)$ coordinates, i.e. $(m-1)\log_2 q$ bits. Thus $\sigma(m-1)\log_2 q$ bits are sent to each server S_j, $1 \leq j \leq q$, for a total of $q\sigma(m-1)\log_2 q$. For his response, each server sends σ field elements for each of the σ points it receives in the query: σ^2 field elements, i.e. $\sigma^2 \log_2 q$ bits, and thus a total of $q\sigma^2 \log_2 q$ bits for all the servers. The overall communication complexity for the queries and the answers is $q\sigma(m-1)\log_2 q + q\sigma^2 \log_2 q = (m-1+\sigma)q\sigma \log_2 q = O(q\sigma^2 \log_2 q)$, since $m \leq \sigma$ as soon as $s > 1$.

Parameters					Locality		Storage overhead		Comm. complexity	
q	m	s	d	k	♯ queries	♯ servers	std	ours	std	ours
16	2	1	14	120	15	16	32	2.1	180	128
16	2	2	29	465	45	16	25	1.7	900	768
16	2	3	44	1035	90	16	22	1.5	2880	2688
16	2	4	59	1830	150	16	21	1.4	7200	7040
16	2	5	74	2850	225	16	20	1.3	15300	15360
16	2	6	89	4095	315	16	20	1.3	28980	29568
16	3	1	14	680	15	16	90	6.0	240	192
16	3	2	29	4960	60	16	50	3.3	1680	1536
16	3	3	44	16215	150	16	38	2.5	7800	7680
16	3	4	59	37820	300	16	32	2.2	27600	28160
16	3	5	74	73150	525	16	29	2.0	79800	82880
16	3	6	89	125580	840	16	27	1.8	198240	207872
16	4	1	14	3060	15	16	320	21	300	256
16	4	2	29	40920	75	16	120	8.0	2700	2560
16	4	3	44	194580	225	16	76	5.1	17100	17280
16	4	4	59	595665	525	16	58	3.9	81900	85120
16	4	5	74	1426425	1050	16	48	3.2	310800	327040
16	4	6	89	2919735	1890	16	42	2.8	982800	1040256
256	2	1	254	32640	255	256	510	2.0	6120	4096
256	2	2	509	130305	765	256	380	1.5	30600	24576
256	2	3	764	292995	1530	256	340	1.3	97920	86016
256	2	4	1019	520710	2550	256	320	1.3	244800	225280
256	2	5	1274	813450	3825	256	310	1.2	520200	491520
256	2	6	1529	1171215	5355	256	300	1.2	985320	946176
256	3	1	254	2796160	255	256	1500	6.0	8160	6144
256	3	2	509	22238720	1020	256	770	3.0	57120	49152
256	3	3	764	74909055	2550	256	570	2.2	265200	245760
256	3	4	1019	177388540	5100	256	480	1.9	938400	901120
256	3	5	1274	346258550	8925	256	430	1.7	2713200	2652160
256	3	6	1529	598100460	14280	256	400	1.6	6740160	6651904
256	4	1	254	180352320	255	256	6100	24	10200	8192
256	4	2	509	2852115840	1275	256	1900	7.5	91800	81920
256	4	3	764	14382538560	3825	256	1100	4.5	581400	552960
256	4	4	1019	45367119105	8925	256	840	3.3	2784600	2723840
256	4	5	1274	110629606725	17850	256	690	2.7	10567200	10465280
256	4	6	1529	229222001295	32130	256	600	2.4	33415200	33288192

Fig. 3. Properties of our scheme for $q = 16$ and $q = 256$. We have to distinguish LDC-locality (i.e. ♯ queries) and PIR-locality (i.e. ♯ servers), since they are not the same using our construction. The storage overhead is the global overhead among all the servers: in the standard case, using the standard LDC to PIR reduction as in Lemma 1, it is $(q-1)/R$; in our case, using partitioning on the servers, it is $1/R$, R being the rate of the code. Similarly the communication complexities (in bits) are shown. The degree d has been chosen to be $d = s(q-1)-1$, the maximum possible value, with no correction capability.

5.3 PIR-locality

Our construction leads to introduce the notion of "PIR-locality": when an LDC code admits a nice layout as multiplicity codes do, the number of servers can be smaller than the locality of the code. We call this the *PIR-locality*. Here the (LDC-)locality, i.e. the number of queries, is $(q-1)\sigma$, while the PIR-locality, i.e. the number of servers, is q. The tables show the obtained parameters for $q = 256$ and $q = 16$ in Fig. 3. We can see that the rate and LDC-locality of the code grow with s, while the PIR-locality is constant for a fixed q. The global storage overhead is much smaller, and the communication complexities are very similar.

5.4 Robustness of the protocol

Algorithm 1 involves σ applications of decoding of univariate multiplicity codes of length $q-1$. From [12], we can decode if the word $y_i = (y_{R_{i,1}}, \ldots, y_{R_{i,q-1}})$ is t-far from a codeword $\mathrm{ev}^s(F)$, for a polynomial $F \in \mathbb{F}_q[X_1]_d$, where $t = 1/2(q-1-d/s)$. The received word y_i corresponds to the answers of the $q-1$ servers (all q servers except server u) for direction $\boldsymbol{U}_i$. Tolerating $\nu = \lfloor t \rfloor$ errors here means that ν servers can answer wrongly. Thus, following the terminology of Beimel and Stahl [3], our protocol is a ν-Byzantine robust protocol.

We sum up features of the protocol presented in Algorithm 2 in the following

Theorem 1. *Let q be a power of a prime, $m, s \in \mathbb{N}^*$, and d be an integer with $d < s(q-1)$. Set $\sigma = \binom{m+s-1}{m}$, with the constraint $\sigma \leq (q^m-1)/(q-1)$. Protocol from Algorithm 2 has:*

- *LDC-locality (i.e. number of queries) $\sigma(q-1)$;*
- *PIR-locality (i.e. number of servers) $\ell = q$;*
- *Communication complexity $(m-1+\sigma)q\sigma \log_2 q$ bits;*
- *Storage overhead $1/R$, where $R = \binom{m+d}{m}/(q^m\sigma)$ is the rate of the underlying multiplicity code;*
- *ν-Byzantine robustness, where $\nu = \lfloor 1/2(q-1-d/s) \rfloor$, in the sense that it can tolerate up to ν servers answering wrongly.*

6 Discussing parameters

6.1 Impact of the Byzantine robustness on the storage overhead

Expressing d in terms of t for a given s gives $d = (q-1)s - 2st$, which then gives a rate, say R_t, to be compared with the rate R found for $d = s(q-1)-1$, when no error can be tolerated. For small m and $s(q-1)$ large enough, we have a relative loss:

$$R_t/R = \frac{\binom{s(q-1)-2st+m}{m}/\sigma q^m}{\binom{s(q-1)+m-1}{m}/\sigma q^m} \approx \left(\frac{(q-1-2t+m/s)}{(q-1+(m-1)/s)}\right)^m$$

For $m = s = 1$, we find $(q - 2t)/(q - 1)$, which is almost the rate of the t-error correcting classical Reed-Solomon code. Otherwise, we get, for small t

$$R_t/R \approx \left(1 - \frac{2t - 1/s}{q - 1 + (m-1)/s}\right)^m$$

For $t = 1$ or 2 and m small, the relative loss is not drastic. But, if t is large, say $(q-1)/2$

$$R_t/R \approx (1/(sq))^m$$

and the loss is bigger.

6.2 Choice of q

We discuss how the size of q may be chosen independently of the size of entries on the database. Consider a simple database, which is a table, with E entries, each entry having S records, all of the same bit-size b. I.e. the total bit-size of the database is thus $N = E \cdot S \cdot b$. A multiplicity code of $\mathbb{F}_q$-dimension k enables to encode $k \log_2 q$ bits. Thus, to encode the whole database, we need $k \log_2 q \geq N = E \cdot S \cdot b$. If furthermore $a = b/\log_2 q$ is an integer, then, to recover a record of size b, the user needs to apply the PIR protocol a times. By definition of information theoretic PIR schemes, Protocol. 2 can be run any number of times, with no information leakage. This implies that q does not need to have a special relationship with the original data.

For instance, imagine a database of 90 000 IPV6 adresses. An IPV6 address consists in 128 bits addresses, i.e. 16 bytes. The database has $E = 90\,000$, $S = 1$, $b = 128$, and requires $90\,000 \cdot 16 = 144\,0000$ bytes of storage. We first design a PIR scheme using $q = 256 = 2^8$. Mapping a byte to an $\mathbb{F}_q$-symbol, we need a code of $\mathbb{F}_q$-dimension at least 144 000. From Table 3, using $m = 3$, $s = 1$, we find a code of $\mathbb{F}_q$-dimension $2796160 \sim 2,7 \cdot 10^6$, and expansion 6. The LDC-locality is 255, and its PIR-locality is 256. The communication cost is 6144 bits.

But we could also use $q_0 = 2^4 = 16$. Then 144 0000 bytes require $2 \cdot 144\,0000 = 2,88 \cdot 10^6$ $\mathbb{F}_{q_0}$-symbols. From Table 3, with $m = 4$ and $s = 6$, we find a code of $\mathbb{F}_{q_0}$-dimension $2919735 \sim 2.9 \cdot 10^6$, and expansion 2.8. Its LDC-locality is $(q_0 - 1)\binom{4+6-1}{4} = 15 \cdot 126 = 1890$ while its PIR-locality is 16. This is better in many aspects since less servers are needed, and a better rate is achieved. But the communication cost is now 1040256 bits.

7 Conclusion

Starting from multiplicity codes, we have designed a layout of the encoded data which leads to a new PIR scheme. It features a very small PIR-locality and much smaller global redundancy compared to PIR schemes naturally arising from LDCs, as well as Byzantine robustness. This layout is quite natural in the context of multiplicity codes. A straightforward question, to be investigated in a future work, is to construct layouts for other locally decodable codes, like affine-invariant codes [8] and matching vector codes [14,6]. This seems feasible due to the very multidimensional and geometric nature of these constructions.

References

1. A. Beimel, Y. Ishai, E. Kushilevitz, and J.-F. Raymond. Breaking the $n^{1/(2k-1)}$ barrier for information-theoretic private information retrieval. In B. Chazelle, editor, *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, volume 59, pages 261–270, 2002.
2. Amos Beimel and Yoav Stahl. Robust information-theoretic private information retrieval. In Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi, editors, *Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 326–341, 2003.
3. Amos Beimel and Yoav Stahl. Robust information-theoretic private information retrieval. *J. Cryptology*, 20(3):295–321, 2007.
4. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, November 1998. Earlier version in FOCS'95.
5. C. Devet, I. Goldberg, and N. Heninger. Optimally robust private information retrieval. In *21st USENIX Security Symposium, Security'12*, pages 269–283, Berkeley, CA, USA, 2012. USENIX Association.
6. Klim Efremenko. 3-query locally decodable codes of subexponential length. In *STOC '09. Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, pages 39–44. ACM, 2009.
7. Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Information Processing Letters*, 43(4):169–174, September 1992.
8. Alan Guo, Swastik Kopparty, and Madhu Sudan. New affine-invariant codes from lifting. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 529–540, New York, NY, USA, 2013. ACM.
9. V. Guruswami and C. Wang. Linear-algebraic list decoding for variants of Reed–Solomon codes. *Information Theory, IEEE Transactions on*, 59(6):3257–3268, June 2013.
10. Brett Hemenway, Rafail Ostrovsky, and Mary Wootters. Local correctability of expander codes. *CoRR*, abs/1304.8129, 2013.
11. Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In F. Yao and E. Luks, editors, *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing, STOC '00*, pages 80–86. ACM, 2000.
12. S. Kopparty, S. Saraf, and S. Yekhanin. High-rate codes with sublinear-time decoding. In Salil Vadhan, editor, *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing, STOC'11*, pages 167–176, New York, NY, USA, 2011. ACM.
13. E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Foundations of Computer Science, 1997. Proceedings. 38th Annual Symposium on*, pages 364–373, October 1997.
14. Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *J. ACM*, 55(1):1:1–1:16, 2008.
15. Sergey Yekhanin. *Locally Decodable Codes and Private Information Retrieval Schemes*. Information security and cryptography. Springer, 2010.
16. Sergey Yekhanin. *Locally Decodable Codes*, volume 6 of *Foundations and Trends in Theoretical Computer Science*. NOW publisher, 2012.

A Possible ranges for d

In order the encoding function ev^s to be injective, it is sufficient to choose $d < sq$. Indeed:

$$\mathrm{ev}^s(f) = \mathrm{ev}^s(g) \Leftrightarrow \mathrm{ev}^s(f-g) = (0, \ldots, 0),$$

which means that $f-g$ admits sq^m zeroes, counting multiplicities. By Schwartz-Zippel lemma, we have:

$$\sum_{P \in \mathbb{F}_q^m} \mathrm{mult}(f-g, P) \leq dq^{m-1}$$

that is here

$$sq^m \leq dq^{m-1}$$

Thus, if we want $f-g$ to be identically zero, it suffices that $d < sq$.

Now during the decoding phase, in the case of errors, one has to perform Reed-Solomon with multiplicities decoding (indeed, σ Reed-Solomon applications of decoding). In this case, the length of the Reed-Solomon code is always $q-1$ as we have $q-1$ noisy evaluations of the original polynomial F on each line. In order such a Reed-Solomon code to realize proper (i.e. injective) encoding, we need $d < s(q-1)$, as shown below.

$$\mathrm{ev}^s(f) = \mathrm{ev}^s(g) \Leftrightarrow \mathrm{ev}^s(f-g) = (0, \ldots, 0),$$

i.e. $f-g$ admits $s(q-1)$ zeroes, where here $\mathrm{ev}^s(f)$ is the encoding of a univariate degree $\leq d$ polynomial $f \in \mathbb{F}_q[X]$ with a Reed-Solomon code of length $q-1$ and multiplicity s. But a univariate polynomial cannot have more zeroes, counted with multiplicities, than its degree:

$$\sum_{P \in \mathbb{F}_q^*} \mathrm{mult}(f-g, P) \leq d,$$

thus if we want $f-g$ to be identically zero, it suffices that $d < s(q-1)$.

B Decoding univariate multiplicity codes

When the number m of variables is 1, then the codes lead to Reed-Solomon codes, also called derivative codes in [9]. We briefly recall how to decode these codes, using the so-called Berlekamp-Welch framework [7]. We consider univariate polynomials in $\mathbb{F}_q[X]$. For $s > 0$, we have $\Sigma = \mathbb{F}_q^s$, and the code is the set of codewords of length $n = q-1$:

$$\left\{ c = \mathrm{ev}^s(F) \mid \ F \in \mathbb{F}_q[X]_d \right\}.$$

Decoding up to distance t is, for a given vector $y \in \Sigma^n$, find all polynomials $F \in \mathbb{F}_q[X]_d$ such that

$$d_\Sigma(\mathrm{ev}^s(F), y) \leq t$$

where d_Σ is the Hamming distance in Σ^n. We first look for two polynomials $N, E \in \mathbb{F}_q[X]$ of degree $(sn+d)/2$ and $(sn-d)/2$ respectively, as follows. Write the linear system of equations:

$$\begin{cases} N(\alpha_i) = E(\alpha_i) \cdot y_{i,0} \\ H(N,1)(\alpha_i) = E(\alpha_i) y_{i,1} + H(E,1)(\alpha_i) \cdot y_{i,0} \\ \vdots \\ H(N,s-1)(\alpha_i) = \sum_{j=0}^{s-1} H(E,j)(\alpha_i) \cdot y_{i,s-1-j} \end{cases}$$

for $i = 1, \ldots, n$, where the indeterminates are the coefficients of N and E. This is a system of sn homogeneous linear equations in $(sn-d)/2+1+(sn+d)/2+1 = sn+2$ unknowns. Thus a non-zero solution (N, E) always exists. Given any solution, F can then be recovered as N/E.

Assuming that $t = (n-d/s)/2$, we can show the correctness of this algorithm: any univariate polynomial F of degree $\leq d$, such that $d_\Sigma(\mathrm{ev}^s(F), y) \leq t$ will satisfy $N - EF = 0$ where (N, E) is a solution of the above system. Indeed, for any α_u such that $\mathrm{ev}^s_{\alpha_u}(F) = y_u$, the system is satisfied at α_u, and hence the polynomial $N - EF$ has a zero of multiplicity s at α_u. Thus

$$\sum_{i=1,\ldots,n} \mathrm{mult}(N - EF, \alpha_i) > (n-t)s = (sn+d)/2.$$

But $\deg(N-EF) \leq \max\{(sn+d)/2, d+(sn-d)/2\} = (sn+d)/2$. Thus $N-EF$, having more zeroes than its degree, is identically zero.